Report No. 415

ILLIAC IV
QUARTERLY PROGRESS REPORT
July, August, and September 1970

Contract No.
USAF 30(602)-4144

ILLIAC IV Document No. 236

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

ILLIAC IV

QUARTERLY PROGRESS REPORT

July, August, and September 1970

Contract No.
USAF 30(602)-4144

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois
61801

October 15, 1970

This work was supported in part by the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, and in part by the Advanced Research Projects Agency as administered by the Rome Air Development Center, under Contract No. USAF 30(602)-4144.

TABLE OF CONTENTS

REPORT SUMMARY

Diagnostics have been produced for all Processing Element (PE) and Memory Logic Unit (MLU) printed circuit card types, including several new and revised types. Diagnostics have been generated for approximately forty percent of those Control Unit (CU) printed circuit cards for which netlists have been received from Burroughs.

The Combinational logic and path tests were maintained and work continued on the control logic phase of the off-line PE diagnostics.

The software effort continues to make good progress. Operating System I (OSI) is virtually complete and has been run successfully on the Burroughs B5500. However, Burroughs' failure to maintain reasonable progress in delivering a satisfactory B6500 MCP and B6500 Compatible ALGOL may cause some delay while the Operating System is translated to B6500 Extended ALGOL. The detailed specification of Operating System II (OSII) is being resumed.

GLYPNIR, Version II, which allows better array handling and I/O facilities, will be released next quarter. Work is well under way in providing a COCKROACH (ILLIAC IV FORTRAN) to GLYPNIR translator for educational and experimental purposes. The Macro-assembler is being consolidated and properly documented.

The PDP-11 System for the Center for Advanced Computation port to the ARPA net is expected to be delivered during the next quarter; much of the software has been designed and implemented using a PDP-11 simulator on the B5500.

Project expenditures and commitments through August, 1970:

Burroughs Corporation	$23,766,784.00
University of Illinois	5,659,542.92

Amendment V to the Burroughs sub-contract is pending approval.

The ILLIAC IV Project Business Office now has an accounting clerk who assists in financial matters and related record keeping.

1. HARDWARE

1.1 Diagnostics

1.1.1 Logic Simulation

1.1.1.1 PE Simulator

The PE Simulator was used principally for verification of the Burroughs functional tests. Some minor discrepancies between the simulator and the actual PE were discovered and corrected.

1.1.1.2 CU Simulator System

Aside from changes to make the generation of boards more trouble-free and the correction of an error in the automatic naming of files, the CU Card Simulator System has remained unchanged.

The compiler for TESLA (the test language used to generate flip-flop initializations), however, has undergone various changes. A new command (FFRAND) has been added to simulate the effect of random settings when the power for a board has been turned on. This aids in determining whether a series of initialization commands will have the proper effect in all cases. The new TESLA compiler also allows automatic file naming, zipping the appropriate board simulator, and more complete documentation of output. Since various bugs have appeared with the more frequent usage of TESLA, a still newer version is under way and should be operational soon.

Two programs (PUNCH/SIGNALS and PUNCH/STORAGE) were written to create decks for running TESLA.

To facilitate the debugging of TESLA, CUTEST/CONTROL, and any other programs which produce simulator input files, a program to

interpret simulator input (SIMIN/PRINTER) was written. It has proved decidedly useful.

Work on the CU Section Simulator continued this quarter. The I-O algorithm for partitioning CU cards [1] was implemented, and a system of programs capable of generating input-part and output-part simulator bodies was completed.

1.1.2 Card Test Generation

1.1.2.1 Card Test Generation System

The major problem which has been encountered in running the test generation system on CU cards is the size of the cards, which causes long program running times and the creation of many large disk files. These effects cause some undesirable conditions.

In order to execute successfully any program which requires more than an hour of continuous processor time, it is necessary that the program be able to recover from HALT/LOADS. This has been done for the test-failure cross reference program (CUFAIL/XREF) and will be done for the LOCATE simulator. A HALT/LOAD proof program to print cross reference dictionaries, which may be up to 3000 pages in length, has been written (CUFAIL/PRINTER). Also, to enable programs which have been zipped but have not yet completed execution to restart execution after a HALT/LOAD, all the programs on the system are being modified so that any program may have a data deck.

Large disk files cause situations in which no additional disk space is available to users, making further processing impossible. Because many of the files simply need to be large, this is a difficult problem to solve completely. Disk allocations on any output files that were found to be over-allocated have been cut down. The most helpful solution, to date, has been to load from the board tape only those files which are needed for the programs to be run. To facilitate this, an

automatic loading program (LOADME/WWOFK) has been written which decides from the contents of the tape which files are to be loaded.

Numerous large files have also caused severe cramps in our supply of backup storage medium (magnetic tapes). Various solutions have been attempted with a certain amount of success. Since library tapes are likely to be only partially used at best (particularly for some of the smaller boards) some of the large files, which previously had their own tapes, have been converted to library files and stored on the library tapes. Two utility programs (CUFAIL/COPY and LOCATE/COPY) have been written to accomplish the storage conversion. At present, nearly all tapes are over 70% utilized. The efficient utilization of tapes has the disadvantage of considerably increasing the amount of time to load or dump files to the tapes. Some thought is presently being expended on ways of automatically removing outdated files.

Since there may be as many as 200 to 300 disk files involved in each board by the time test generation is completed, organizational problems can result. A program to print compact and alphabetized tape directory listings (TAPEDIR/WWOFK) has proved valuable in counteracting this problem.

1.1.2.2 Card Test Translation System

Using the experience gained from earlier versions, the final version of the CU Card Test Translation System was implemented and debugged early in the quarter. The new version emphasizes economical usage of both real and process time during its operation. The new, extensively revised, XALGOL version of the PEX compiler proved itself far faster and more efficient than its predecessor. The PEXTAP translators for both the main-programs and initializations were revised to produce in-line code, eliminating hundreds of slow procedure calls and making full use of the eight microsequence memory registers in the exerciser. Consequently, both translation and compilation are

30% faster, with an average process-to-real time ratio of 6:10. The INIT program can be executed from a teletype which speeds creation of initialization code; for example, 50 INITs in 15 minutes.

Wrap-around test procedures in the form of two pseudo-boards, TEST1 and TEST2, were created to test the reliability of the CU Card Tester. This was accomplished by writing two netlists in which all pins were used (half as input and half as output), with each input logically connected to an output pin. Any discrepancies in testing will reflect malfunctions and/or errors in the yet virgin CU Card Tester Exerciser.

1.1.2.3 PE Card Test Generation

Seventeen new or revised PE card logic prints were received at the beginning of the quarter. Tests for these were quickly generated and forwarded to Burroughs.

During the last week of the quarter thirteen more PE card revisions were received. Tests for these will be generated during the first few days of the new quarter. In addition, an invalid test sequence was generated for the A16 card and must be corrected.

1.1.2.4 CU Card Test Generation

Due to personnel changeover, vacations, and machine down-time, test generation fell considerably behind projections this quarter. Nevertheless, the following processing has taken place on the sixty-six board types whose netlists have been received from Burroughs:

Board Simulators Generated	60
Failure Detection Complete	31
Test Tapes Generated	27
Dictionaries Generated	30
Test Tapes Sent to Burroughs	26
Dictionaries Sent to Burroughs	23

The continuing lack of feed-back, due to the non-functioning of the Burroughs CU Card Tester for the entire past quarter, is disheartening.

1.1.3 PE Diagnostics

1.1.3.1 Path Tests

In this quarter the path tests were maintained. The updated path tests have been sent to Burroughs.

1.1.3.2 Combinational Tests

The combinational tests were completed during this quarter. The test sequences have been sent to Burroughs.

1.1.3.3 Control Logic Tests

The control logic test is divided into three main tasks: equation generation, test pattern generation for the logic being tested, and test sequence generation with respect to the PEX and PE.

Updating of the equations is 90% complete. The completion of this first task requires the latest version of the PE backplane netlist which has not been received from Burroughs.

The second task is complete. The algorithm used for test generation is new and unique. The completeness of its fault location capability was reviewed and the study revealed some flaws in the algorithm. A new one has, therefore, been derived.

A file editing program has been written for obtaining the control logic test sequences in PE exerciser assembly language. Initializations for the tests, test data inputs, and expected responses are generated manually. Progress was impaired by a lack of knowledge of the design details of the PE, unavailability of the latest backplane netlist, and inconsistencies between logic diagrams and wire

lists. Greater progress is expected in the future following a better acquaintance with the PE design.

1.1.3.4 Functional Tests

Verification of all but a couple of the Burroughs functional tests has been completed using the PE Simulator. Almost all functional tests were found to be correct. Those containing errors were reported to Burroughs.

2. SOFTWARE

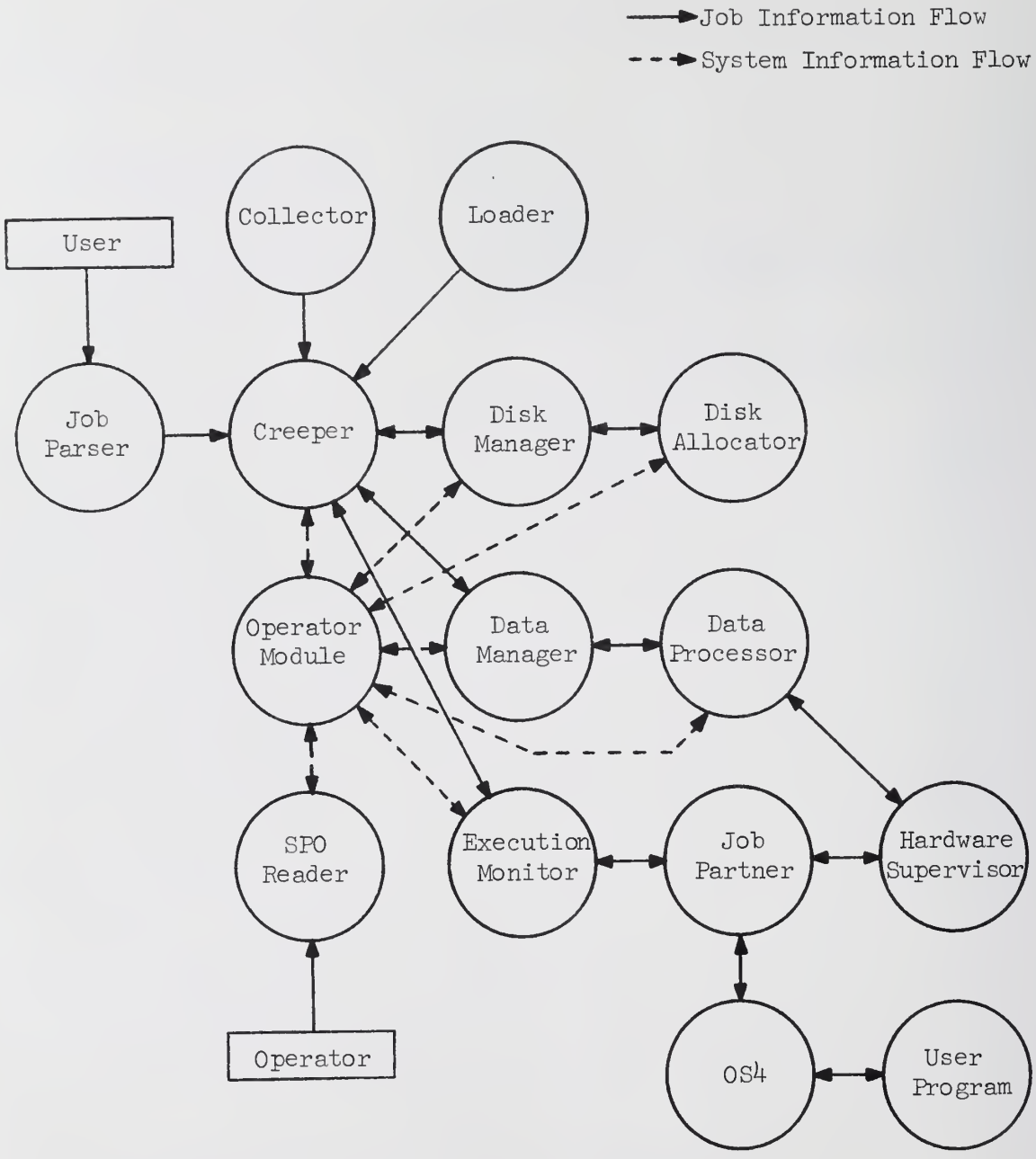
2.1 Operating System

2.1.1 Operating System I

All modules (see following page) of the operating system except the Job Partner, Hardware Supervisor, and OS4 (that part of the operating system which resides in the ILLIAC IV) are coded, debugged, integrated, and running as a unity on the B5500. A Simulated Hardware Supervisor is being provided for the B5500 integration which will allow the completion of the Job Partner, and the running of OS4 (under the ILLIAC IV simulator) with this integrated system. OS4 has been held up because of the recent discovery of undocumented changes in the interrupt system, which necessitated its rewriting. The Hardware Supervisor is being delayed by the poor reliability of the B6500 and its software. These delays, in turn, prevent the completion of the Job Partner, in the light of Burroughs' statements that Compatible ALGOL will have the lowest priority as far as completion and maintenance is concerned; further delays will be necessitated while the entire operating system is translated from Compatible to Extended ALGOL. Despite these delays, the entire operating system should be integrated and working with the ILLIAC IV simulator by the end of November.

Work is proceeding with the following enhancements to the operating system:

- enhanced communication over the 48-bit path between the B6500 and ILLIAC IV which will allow more diverse control of the system by ILLIAC IV;



Operating System I: Module Configuration

- adding timing and other instrumentation to the operating system (there was none in the original design), to help with its tuning in simulation or with the real ILLIAC IV;
- augmented version of break facilities to help with debugging and scheduling;
- an investigation of scheduling algorithms to ILLIAC IV, including a timing simulation of ILLIAC IV;
- disk map checking facilities in the ILLIAC Control Language, to insure that the disk mapping requested by the user is realizable and to inform him if it is not so before his job execution is commenced.

User, System, and Maintenance documentation is being prepared.

2.2 Compilers and Translators

2.2.1 GLYPNIR

Version II of GLYPNIR will be released very shortly. The major enhancements (compared to Version I) are:

- I/O statements for:
  a) semi-formatted I/O statements for use with the ILLIAC IV Simulator,
  b) block oriented, unbuffered disk I/O;
- vectors as a data type (previously, vectors had to be accessed with pointers);
- improved interface with assembly language, allowing easier embedding of Maçro-assembler code within GLYPNIR code;

· more extensive standard mathematical routines and intrinsics for routing, rotating, and shifting.

2.3 Macro-Assembler

The Macro-assembler, as it stands, evolved from the assembler, and, as a consequence, its workings are rather fortuitous. Thus, despite the fact that most users are finding it satisfactory, some bugs occurring in the "evolutionary" part of the Macro-assembler are becoming very difficult to find and correct. This part of the Macro-assembler is thus being re-designed and rewritten. The re-write should be finished by the end of October. It will not affect running programs.

An effort to collect and compile a unified, coherent Macro-assembler Manual from the scattered literature, now available, continues.

2.4 Interactive Communications and Graphics

When it became apparent that the ILLIAC IV might not be located on the Urbana Campus of the University of Illinois, steps were taken to ensure that both the University of Illinois and Burroughs (Paoli) were linked to the ARPA Net. This involves interfacing an IMP to the B6500 at Paoli and an IMP to a PDP-11 at Urbana. Limited graphics facilities will be interfaced to the PDP-11; but the main function of the PDP-11 is to handle ARPA Net and user protocol for the Center. In order to make the best of the limited time available, a PDP-11 simulator, assembler, and a higher level language with which to program the operating system were started. It is anticipated that the PDP-11 will be delivered by the end of 1970 and that the link to Paoli will be operational by April 1971.

- Hardware Status
  - · the basic system design around the PDP-11 has been completed;
  - · the PDP-11 System bid is in progress; the system is expected to be ordered on October 15, 1970;

- a graphics terminal has been ordered;
- a PDP-11 to IMP interface has been designed, checked, and is under construction;
- the B6500 to IMP interface proposal was returned to Paoli on August 18, 1970 for reconsideration by Burroughs;
- hardware configuration checked at Digital Equipment Corporation August 31, 1970.

• Software Status

- preliminary investigations are under way into B6500 MCP code changes for data-communications;
- PDP-11 System, Version I, is 50% coded;
- a PDP-11 assembler on the B5500 is working and is being extended;
- a PDP-11 simulator on the B5500 is being coded and debugged;
- a single contour line generation routine for ILLIAC IV is being completed and checked;
- a surface presentation routine for ILLIAC IV is coded;
- a Calcomp type plot system for the terminal is being designed.

2.5 Debugging Package

Work on the debugging package has progressed to the stage where a simple version will shortly be made available with the ILLIAC IV simulator on the B5500. Further information is presented in FOURUM Communication No. 26.

2.6 B5500 Operation

	No. of Jobs	Process Hours
July	25,989	180.85
August	19,330	208.62

A few days were lost during the quarter due to air-conditioning problems; these are now solved. There has been a hardware problem with the memory mod files; although it has been fixed, memory checks still occasionally appear--a HALT/LOAD usually restores the machine.

Throughput has been good and response time has been fine. The present status of the MCP is Mark X, LEVEL 36.43.

3. APPLICATIONS

3.1 Numerical Analysis

3.1.1 Matrix Inversion and Solution of Linear Algebraic Equations

This quarter, the method of Householder triangularization as described in the last QPR [2] was coded in ASK and executed on the SSK simulator. This program, which consists of several routines, inverts a matrix or solves a matrix equation, AX = B, where A, B, and X are matrices.

The task at hand is to provide an iterative improvement scheme using the results of the above program. Two routines are coded in ASK: one for improvement of the solution of the matrix equation, and another for improving the inverse of a matrix. The former routine is in a check-out phase on the simulator, and the latter routine, the improvement for the inverse of a matrix, is executing correctly on the simulator.

The procedure for the improvement of the solution is as follows: If X_0 is the result obtained from Householder triangularization, the residual vector or matrix is obtained,

$$R_m = B - AX_m \quad .$$

The correction matrix, S_{xm}, can be found by solving the equation

$$AS_{xm} = R_m \quad .$$

Provided that double precision arithmetic is not used and that the upper triangle and other requisite vectors are saved from the original

reduction, S_{xm} is readily obtained by back-substitution. The improved solution becomes:

$$X_{m+1} = X_m + S_{xm} \quad .$$

The improvement of the inverse involves a similar procedure--without the necessity of solving a matrix equation at each iteration. If X_0 is the result of Householder triangularization, find

$$D_0 = I - AX_0 \quad .$$

Obtaining D_0, the improved inverse may be derived as:

$$X_m = X_{m-1} (I + D_{m-1}) \quad ,$$

where $D_m = D_0^{2m}$.

3.1.2 Eigenvalue Problems

3.1.2.1 Jacobi's Algorithm

Jacobi's algorithm for finding eigenvalues of symmetric matrices has been written and debugged and is in the stage of being tested for different-size matrices. A document describing the algorithm and the features of the program, written in ASK, is being written.

3.1.2.2 Eberlein's Algorithm

A Jacobi-like algorithm for real non-symmetric matrices is under investigation. A short description of the algorithm [3] may be presented as: "A matrix, A, is normalized by undergoing a sequence of orthogonal transformations. Once the matrix is normalized (i.e., the matrix $C = (A \cdot A^T - A^T A)$ is arbitrarily small) one is able to reduce the matrix to a diagonal form and, thus, obtain the eigenvalues of A."

The orthogonal transformation matrices are of the form:

$$(1) \qquad M_\ell = \begin{cases} (I_{1u} \cdot I_{2v}) & (u, v) \text{ different from } (2k\text{-}1, 2k)\ \forall k \\ I & \text{otherwise} \end{cases}$$

This transformation is used to speed up the rate of convergence.

(u,v) is found by searching for $\max|c_{ij}|(i \neq j)$. The transformation brings the element, c_{ij}, thus found, into the position (2k-1, 2k), where it will have the most appreciable effect on the convergence of the algorithm.

$$(2) \qquad P_\ell = \text{diag}\,(T_1^{(\ell)}, T_2^{(\ell)}, \ldots, T_{n/2}^{(\ell)})$$

$$\text{with} \qquad T_k = \begin{bmatrix} \cos \varphi_k^{(\ell)} & \sin \varphi_k^{(\ell)} \\ -\sin \varphi_k^{(\ell)} & \cos \varphi_k^{(\ell)} \end{bmatrix} .$$

$$\varphi_k \text{ is determined by } \tan 2\varphi_k^{(\ell)} = \frac{c_{2k-1,2k-1}^{(\ell)} - c_{2k,2k}^{(\ell)}}{2c_{2k-1,2k}^{(\ell)}} .$$

The orthogonal matrix, P, is chosen such that the off-diagonal elements, $c_{2k-1,2k}$, for all k attain their maximum positive value.

$$(3) \qquad Q_\ell = \text{diag}\,(S_1^{(\ell)}, S_2^{(\ell)}, \ldots, S_{n/2}^{(\ell)})$$

$$\text{with} \qquad S_1^{(\ell)} = S_2^{(\ell)} = \ldots = S_{n/2}^{(\ell)} = \begin{bmatrix} \cosh \psi_\ell & \sinh \psi_\ell \\ \sinh \psi_\ell & \cosh \psi_\ell \end{bmatrix} .$$

ψ_ℓ is derived from:

$$\tanh 4\psi_\ell = -2k_2(A'_\ell)\, k_1(A'_\ell) \quad ,$$

where

$$A'_\ell = (M_\ell P_\ell)^T A_\ell (M_\ell P_\ell) \quad ,$$

$$k_2 = \sum_{k,m} D_{km} E_{km} \text{ and } k_1 = \sum_{k,m} (D_{km}^2 + E_{km}^2)$$

with

$$D_{km} = (a_{2k-1,2m-1} - a_{2k,2m}) \text{ and } E_{km} = (a_{2k-1,2m} - a_{2k,2m-1}) \, .$$

Where (1), (2) and (3) are applied to $A_{\ell+1}$, i.e.,

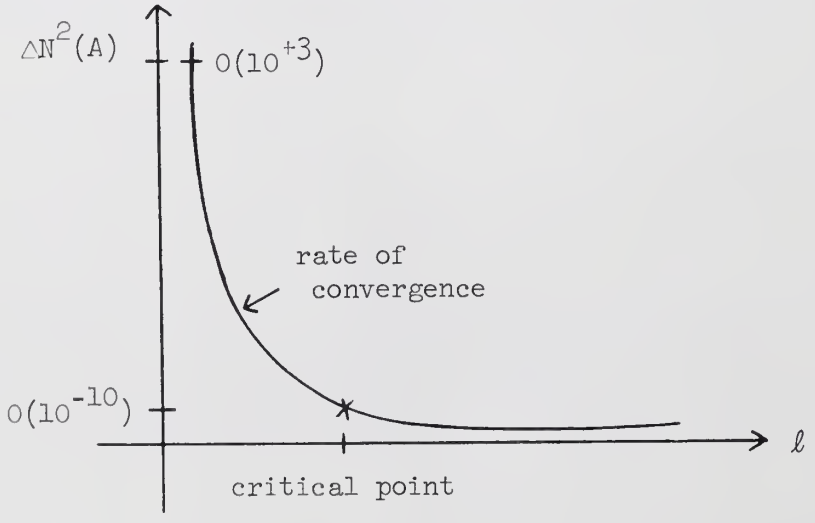
$$A_{\ell+1} = (M_\ell P_\ell Q_\ell)^{-1} A_\ell (M_\ell P_\ell Q_\ell) \quad ,$$

$$\Delta N^2(A_\ell) \to 0 \text{ as } \ell \to \infty, \text{ hence } N^2(C_\ell) \to 0 \quad ,$$

A_ℓ is arbitrarily close to being normal.

The following tests were undertaken in an attempt to find when A is close to being normal for a finite ℓ.

Tests showed that when plotting $\Delta N^2(A)$ against ℓ, the rate of convergence levels off at $\Delta N^2(A) = 0(10^{-10})$ rather quickly, but the matrix, A, does not show satisfactory normalization.



Tests also showed that at $\Delta N^2(A) = O(10^{-10})$ the elements of c and, especially, $\max|c_{ij}|$ begin to oscillate as soon as they reach the order of $\epsilon = 10^{-5}$. This oscillation is critically reflected in the transformation matrix, P, which is close to the identity-matrix at that point and, therefore, does not contribute significant changes in A toward normalization.

A suggestion which might solve the problem is to check

$$F = |c_{2k-1,2k-1} - c_{2k,2k}| \leq 10^{-b}, \text{ where b ranges from 1-3.}$$

If this test is satisfied, then Φ_k should not be determined by using $F \neq 0$, but rather by using the Φ_k for $F = 0$ [3]. Investigation for determining a most general b and its effect on the matrix under consideration is under way--early indications show promising results.

3.1.2.3 QR-Algorithm

The ASK program for finding eigenvalues of up to 64 x 64 real Hessenberg matrices using the QR-algorithm [4] has been written. The program is currently in the final stage of being debugged on the SSK simulator; it is expected to be complete by the end of October.

3.2 Linear Programming

Assembly and simulation of segments of the Linear Programming System (LPS) code continues as a major task of the linear programming group. The LP project has been allocated one hour each day of priority use of the B5500 computer for this purpose. During this time an average of approximately 2 ASK compilations and 2 SSK simulated executions can be run. The inclusion of macro instructions in the ASK code stream significantly increases the compile time over that of non-macro codes. Most of the LP codes now contain the macro I/O instructions discussed in the previous QPR, and thus computer throughput has been reduced.

Software efforts of the LP group have included debugging the LP I/O macros and developing subroutine linkage conventions for ASK applications programs. User documentation for the I/O macros has been provided.

Code development for the LP matrix reinversion routine, INVERT, was begun this quarter. A modular system flowchart was drawn up for the inversion procedure discussed in "An Error Analysis of Solutions to Sparse Linear Programming Problems" by R. Jonathan Lermit [5]. Many modifications had to be made in the implementation design of this algorithm. Detailed flowcharts and ILLIAC IV assembly code were produced for the initial data processing modules of the INVERT routine.

At a recent meeting, the 7th Mathematical Programming Symposium held in the Netherlands, E. Hellerman and D. Ranck presented a paper entitled, "Reinversion with the Preassigned Pivot Procedure", outlining an LP inversion technique which produces an improved inverse. It is claimed that implementation of this scheme for the CDC and IBM LP systems performs reinversion six to ten times faster than the previous procedure and substantially reduces the number of non-zeros in the inverse matrix file. In discussions with one of the authors, it was determined that this scheme is suitable for the ILLIAC IV LP system. The code already produced for the previous INVERT design is still applicable; new system module flowcharts will be developed during the next quarter.

3.3 Long Codes

Following an extensive revision to provide better documentation, the system identification programs were used to study the ability of the identification algorithms to estimate the parameters of linear dynamic systems under a variety of conditions. In order to study the effects of varying levels of noise contamination, the variance of the essentially Gaussian noise vectors added to the system state vectors (plant noise) and the variance of the noise vectors added to the observation vectors (observation noise) were independently varied by

scaling the noise vectors used. The observed vector for a given time step should be affected by the observation noise vector occurring at the current time step only, while plant noise vectors from previous time steps should affect the observation through the transition matrix. Since it is desirable to have the noise terms for a given time step uncorrelated with the noise from previous time steps, plant noise was expected to have a more detrimental effect on the accuracy of the identification process than observation noise. The experiments conducted, however, showed the reverse is often true. To cope with this, a new algorithm specifically designed to minimize observation noise contamination has been developed and is being implemented. Studies were also begun to determine the ability of the identification algorithms to identify systems formed by subjecting transformation matrices in the more desirable companion matrix form to arbitrary non-singular similarity transformations.

3.4 Large Scale Planning

Much of the work in this area during the past quarter has been concentrated on the geometric matrix generator mentioned in the last QPR [2]. The rationale for this generator, its preliminary specification, and a brief discussion of a proposed report generator follow.

3.4.1 Matrix Generator

Since linear programming matrices are typically very sparse and contain repetitive, dense sub-arrays of non-zero coefficients, they lend themselves to formulation by the use of matrix-generation codes. There are several possible approaches to the problem of matrix generation, including model-specific generators, generators based on algebraic functions relating variables, and generation languages. The approach to be used here is that of language. Since the number of geometric structures which occur as sub-arrays within a linear programming matrix is small, the generator language is designed to allow the user to define

sub-arrays in geometric terms, and build a matrix for a specific problem by providing dimensions and coefficients for these sub-arrays.

The structures occurring in linear programming matrices are:

(i) point;
(ii) row;
(iii) column;
(iv) square;
(v) rectangle;
(vi) parallelogram;
(vii) diagonal;
(viii) band;
(ix) lower band;
(x) upper band;
(xi) lower triangle; and
(xii) upper triangle.

Structures (vi) through (xii) have left and right mirror images, which are themselves structures. Also, a band matrix is an extension of a diagonal and can be represented as 5-diagonal, 3-diagonal, etc. "Upper band" and "lower band" are structures in which exists the main diagonal with one or more secondary diagonal(s) above or below it.

Since these structures, in combination, form the structure of a sparse matrix, their existence can be exploited to form a linear programming matrix. This is the philosophy of the matrix generator.

In order to use this principle to create a matrix, it is necessary to define the structure required for a sub-array, its location in the matrix, its dimensions, the coefficients to be inserted, and the number of times the structure is repeated across the matrix.

Sub-array location in the matrix is defined by an (x,y) coordinate system, with (0,0) at the northwest corner of the matrix. This coordinate system is also used to identify the blocks of columns and rows to which the specific real-world activities and constraints

relate. (User-generated real names can be input and passed to the report generator for solution interpretation.)

The characteristics of the structure definitions are:

(i) POINT
- Size - none given
- Value - only one number
- Repeat Factor - single element repeated horizontally by number given as repeat factor

(ii) ROW
- Size - one number gives number of columns spanned
- Value - values given are accepted, one per column. If the number of values is less than the given size, the last value is repeated for the remaining columns.
- Repeat Factor - row structure defined is repeated specified number of times

(iii) COLUMN
Same as ROW, mutatis mutandis

(iv) SQUARE
- Size - one number taken as the number of rows and columns
- Value - only one number given, which is assigned to all S^2 elements
- Repeat Factor - horizontal repetition of the S^2 substructure

(v) RECTANGLE
- Size - two numbers are accepted, with c referring to columns and r to rows
- Value - one number is assigned to all cr elements
- Repeat Factor - horizontal repetition of cr-element substructure

(vi) PARALLELOGRAM [Left and Right]
- Size - two numbers, the first refers to number of elements in each diagonal and the second to the number of diagonals
- Value - number of values equal to the number of diagonals accepted, the first referring to the uppermost and the others to lower diagonals, in order
- Repeat Factor - horizontal repetition

(vii) DIAGONAL [Left and Right]
Same as ROW, mutatis mutandis

(viii) BAND [Left and Right]

Size - two numbers accepted; the numbers of diagonals, b, and the number of elements on the main diagonal, d. The number of elements on adjacent subdiagonals is d - 1.

Value - as for PARALLELOGRAM

Repeat Factor - matrix of size d × d with banded structure is repeated horizontally

(ix) LOBAND [Left and Right]

Same as BAND

(x) HIBAND [Left and Right]

Same as BAND

(xi) LOTRI [Left and Right]

Size - one number, d, is taken as the size of a square submatrix

Value - each number given is used as the value for all elements of one lower triangular matrix. A sequence of numbers will define an equal number of lower triangular matrices, with entries equal to the numbers in the sequence.

Repeat Factor - the sequence of lower triangular matrices defined by the given values are repeated horizontally

(xii) HITRI [Left and Right]

Same as LOTRI

The matrix generator will permit the user to structure his model using declarations giving the starting coordinates, the sub-structure, its dimensions, the value(s) to be inserted, and the repeat factor. For example, the statement

```
50,100;LOTRI(50):1.0*10
```

causes the generation of a 50 by 50 left-lower triangular matrix of values 1.0, starting at row 50 and column 100, and repeated 10 times horizontally. This one statement thus generates 12,750 matrix coefficients which would have been entered, using conventional means, as 12,750 entries <ROW NO. COLUMN NO. VALUE>. Similarly, the statement

400,1000;COL(12):LABORA*50

will create 50 contiguous columns of 12 values drawn from a file named LABORA, starting at row 400 and column 1000. The conventional procedures would have required 600 card entries to input these data.

3.4.2 Report Generator

A major source of the costs of using linear programming is that of interpretation of solutions. In the conventional use of linear programming the solution output, consisting of lists of basis vectors and details concerning them, must be manually interpreted. When the model is of the magnitude of a sectoral model, the task of interpreting the solution is formidable and costly. The objective of a report generation system is to facilitate the output of interpretive reports of the solution to a linear programming problem.

The report generator to be developed here is closely related to the matrix generator discussed above. The real-world names of activities and constraints will be defined by the user as input to the matrix generator. The sets of columns and rows spanned by these names will be given dimensions in terms of the (x,y) coordinate system used by the matrix generator. This will enable the report generator to match the real-world names with the column and row names internally assigned by the linear programming code. Report formats will be variable and will allow some user-defined variations dependent on the requirements of the decision makers receiving the output. This will be accomplished by allowing the user to specify the sets of linear programming output to be passed through the report generator. For example, if the user desires the marginal products of certain limiting resources, he will be able to specify these by an input declaration which will cause the linear programming "reduced costs" to be output in the report.

3.4.3 Sector Model

The generalized sector model is also in the definitional phase. The philosophy being followed is that of generic activity and constraint

sets which can be reproduced by the matrix generator under parameter control. The next report will discuss the model philosophy in more detail.

3.5 Signal Processing

The matrix convolution program is being extended to handle matrices with column dimensions in excess of 64. It is anticipated that this effort will be completed and documented in the first half of the fourth quarter. The program, written in ASK, currently is operational for the convolution of two matrices whose resulting matrix has 64 columns or less.

A preprocessor program to read 9-track input tapes and set up tables and file allocations for all line, shot, velocity and normal move-out information is approximately 25% complete. This program is being written for the B6500 computer to organize the data for operations in the ILLIAC IV, utilizing ILLIAC IV as: 1) a group of 64 serial computers; 2) a group of 128 serial computers using 32 bit mode; and 3) using the parallel concepts developed in the signal processing programs previously written as well as those planned for future development.

3.6 ILLIAC IV Education

3.6.1 Education

3.6.1.1 ILLIAC IV Text

The initial version of the ILLIAC IV textbook to be titled "An Introductory Description of the ILLIAC IV System" will be issued next quarter. Updated material will be sent to the FOURUM members who request copies of the book. Thus the distribution will be handled like that for the "Systems Characteristics and Programming Manual."

3.6.1.2 ILLIAC IV Seminars

Following is the outline of a two-day seminar on ILLIAC IV that was held on August 24th and 25th. Two more of these seminars are planned for next quarter.

August 24, 1970

Background	1 Hour	9:00 - 10:00

Functional Components of a Computer

Buffer Concept -- IBM 7090, Overlap of Operations, Example

Pipeline Concept -- Pipelined Adder, Pipelined Instruction Execution, IBM STRETCH

Array Processor -- SOLOMON

Multiprocessors -- Multiprogramming and Multiprocessing, General Multiprocessor, Intrinsic Multiprocessor

ILLIAC IV

Hardware Structure	5 Hours	10:00 - 12:00 2:00 - 5:00

ILLIAC IV Array -- General Description, Two Illustrative Problems, Data Allocation Considerations, A More Refined Description

ILLIAC IV Input/Output (I/O) System -- I/O Subsystem, Disk File System, B6500 Control Computer, Datacommunications Processor

Test/Repair Equipment and Diagnostics

August 25, 1970

Software	5 Hours	9:00 - 12:00 2:00 - 4:00

Programming Languages -- ASK, GLYPNIR, FORTRAN

Operating System

Utilities

Capabilities and Applications	1 Hour	4:00 - 5:00

3.6.1.3 CS-491-B

The graduate course "Architecture, Applications, and Languages for a Parallel Computer" (Computer Science 491-B) is being offered again this fall. The outline follows:

I. Background — 8 classes

- A. Functional Components of a Computer
- B. Buffer Concept
  - 1. Overlap of Operations
  - 2. Example
- C. Pipeline Concept
  - 1. Pipelined Adder
  - 2. Pipelined Instruction Execution
- D. Array Processor
  - 1. SOLOMON
- E. Multiprocessors
  - 1. Multiprogramming and Multiprocessing
  - 2. General Multiprocessor
  - 3. Intrinsic Multiprocessor
- F. ILLIAC IV

II. Hardware Structure — 14 classes

- A. ILLIAC IV Array
  - 1. General Description
  - 2. Two Illustrative Problems
    - a. Summing an Array of Numbers
    - b. Relaxation Solution to Laplace's Equation in Two Dimensions
  - 3. Data Allocation Considerations
  - 4. A More Refined Description
    - a. Processing Element
    - b. Processing Unit
      - i. Processing Element Memory (PEM)
      - ii. Memory Logic Unit (MLU)
    - c. Control Unit
      - i. Advanced Station (ADVAST)
      - ii. Final Station (FINST)
        - (a) Final Queue (FINQ)
          - (1) Final Data Queue (FDQ)
          - (2) Final Instruction Queue (FIQ)
      - iii. Memory Service Unit (MSU)
      - iv. Test and Maintenance Unit (TMU)
      - v. Instruction Look Ahead (ILA)

B. ILLIAC IV Input/Output (I/O) System
   1. I/O Subsystem
      a. Control Descriptor Controller (CDC)
      b. Input/Output Switch (IOS)
      c. Buffer Input/Output Memory (BIOM)
   2. Disk File System
   3. B6500 Control Computer
      a. CPU
      b. Memory
      c. Multiplexor
      d. Peripherals
   4. Datacommunications Processor
      a. Remote Terminals
      b. Interactive Graphics
      c. Microfilm Printer
      d. Laser Memory
      e. ARPA Network

C. Test/Repair Equipment and Diagnostics
   1. Processing Element Exerciser (PEX)
   2. Processing Element Memory Exerciser (PEMX)
   3. Control Unit Card Tester (CUCT)
   4. Diagnostics

III. Software — 15 classes

A. Operating System
   1. General Philosophy
   2. ILLIAC IV from the Users Point of View
   3. Details of Operating System
      a. Burroughs B6500 MCP
      b. Job Parser
      c. The Creeper
      d. Disk Management
      e. Collector and Loader
      f. Job Partner
      g. Operating System IV
      h. Data Processor on Other Modules
   4. Implementation of the Operating System

B. Programming Languages
   1. General Philosophy and a Brief History
   2. The Assembler System
      a. Assembler
      b. Macro Generator
      c. Simulator
   3. Intermediate Languages
      a. GLYPNIR
      b. FORTRAN
   4. Higher Level Languages
      a. Experience with TRANQUIL
      b. APL
   5. Implementation Techniques

C. Utilities
   1. Data Communications Software
   2. Program Debugging Software
   3. Statistical Package
   4. Linear Programming Package

IV. Capabilities and Applications 3 classes

3.6.2 Documentation

3.6.2.1 Quarterly Progress Report

The distribution list of the ILLIAC IV QPR has been effectively decreased by approximately 200 people as the result of a "weeding" letter to QPR recipients.

3.6.2.2 FOURUM

Requests for research documents have been steadily received from FOURUM subscribers since the publication of the July issue of FOURUM.

One hundred documents per month have been requested by FOURUM members (outside the University of Illinois).

4. ADMINISTRATION

4.1 Administration and Services

4.1.1 Financial Report

Actual expenditures and commitments for July and August, 1970:

	July	August
Burroughs Corporation	$714,784.00	$431,000.00
University of Illinois	210,074.86	198,097.02

Expenditures and commitments to date--through August, 1970:

Burroughs	$23,766,784.00
University	5,659,542.92

Budgeted expenditures--First Quarter, Fiscal Year 1971:

	July	August	September
Burroughs	$244,630.00	$239,078.00	$167,502.00
University	195,398.00	195,398.00	195,398.00

4.1.2 Personnel Report

Project personnel strength:

Professional	42 1/4
Non-Academic	18
Research Assistants	26
Hourlies	24
Illini Girls (Secretarial Assistants)	7
	117 1/4

The ILLIAC IV Project Business Office now has an accounting clerk who assists in financial matters and related record keeping.

4.1.3 Contract Status

Amendment V to the Burroughs sub-contract is pending approval.

The building designed to house the Center for Advanced Computation is within approximately thirty days of the originally scheduled occupancy date of January, 1971.

REFERENCES

[1] A. Hashimoto, L. Abel, and C. Tanaka, "Improvement of Section Simulation Efficiency by Partitioning Board Simulators," ILLIAC IV Document No. 216, DCS File No. 836, (May 8, 1970).

[2] ILLIAC IV Quarterly Progress Report, April, May, and June 1970, ILLIAC IV Document No. 224, DCS Report No. 408, (July 15, 1970).

[3] A. Sameh, "On Jacobi and Jacobi-Like Algorithms for a Parallel Computer," ILLIAC IV Document No. 227, DCS File No. 852, (to be published).

[4] R. S. Martin, G. Peters, and J. H. Wilkinson, "The QR Algorithm for Real Hessenberg Matrices," Numer. Math. 14, 219-231, 1970.

[5] R. J. Lermit, "An Error Analysis of Solutions to Sparse Linear Programming Problems," ILLIAC IV Document No. 192, DCS Report No. 351, (September 17, 1969).

THESES

Inagaki, Masayuki, "Diagnosis of Printed Circuit Cards on a Programmable Test Equipment." Master's thesis, ILLIAC IV Document No. 228, DCS Report No. 211. Urbana, Illinois: Department of Computer Science, University of Illinois at Urbana-Champaign, 1970.

DOCUMENTS

Funches, Jesse, "Cross Correlation: A Computer Program," ILLIAC IV Document No. 219, DCS File No. 845, (June 15, 1970).

Lai, Steven, "Correlation Analysis Program," ILLIAC IV Document No. 221, DCS File No. 847, (June 15, 1970).

Serage, Jeffrey, "Sample ASK Programs," ILLIAC IV Document No. 220, DCS File No. 846, (June 15, 1970).

Stevens, James E., Jr., "A Fast Fourier Transform Subroutine for ILLIAC IV," ILLIAC IV Document No. 226, DCS File No. 851, (July 28, 1970).

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP

3. REPORT TITLE
ILLIAC IV QUARTERLY PROGRESS REPORT
July, August, September 1970

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
July-September 1970 - Progress Report of the ILLIAC IV Project

5. AUTHOR(S) *(First name, middle initial, last name)*

6. REPORT DATE
October 15, 1970

7a. TOTAL NO. OF PAGES
39

7b. NO. OF REFS
5

8a. CONTRACT OR GRANT NO.
USAF 30(602)-4144

b. PROJECT NO.
46-26-15-305

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
ILLIAC IV Document No. 236
DCS Report No. 415

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
RADC TR

10. DISTRIBUTION STATEMENT
Qualified requesters may obtain copies of this report from DCS.

11. SUPPLEMENTARY NOTES
None

12. SPONSORING MILITARY ACTIVITY
Rome Air Development Center
Griffiss Air Force Base
Rome, New York 13440

13. ABSTRACT

See the Report Summary on Page 1 within the Report itself.

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Design and Construction						
Components and Circuits						
Hardware Diagnostics						
Compilers and Generators						
GLYPNIR						
Graphics						
Numerical Analysis						
Linear Programming						
Operating System I						
Assembler						
Ordinary and Partial Differential Equations						
Linear Algebra						
Matrix Algebra						
Education						
Simulators						
Signal Processing						
Utility Programs						
Administration of Computing Centers						